

**PATENT**

**MS146953.01/MSFTP250US**

**CERTIFICATE OF TRANSMISSION**

I hereby certify that this correspondence (along with any paper referred to as being attached or enclosed) is being submitted *via* the USPTO EFS Filing System; Mail Stop Amendment; Commissioner for Patents; P.O. Box 1450; Alexandria, VA 22313-1450.

February 28, 2007  
Date:

/Casey Martin/  
Casey Martin

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re patent application of:

Applicant(s): Daniel Dedu-Constantin, *et al.*

Examiner: Te Y. Chen

Serial No: 09/894,653

Art Unit: 2161

Filing Date: June 27, 2001

Title: SYSTEM AND METHOD FACILITATING UNIFIED FRAMEWORK FOR  
STRUCTURED/UNSTRUCTURED DATA

**Mail Stop Amendment  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450**

---

**DECLARATION UNDER 37 C.F.R. §1.131**

---

Dear Sir:

The undersigned declare as follows:

- 1) We are the inventors of the above-identified patent application.

2) At least as early as April 11, 2001 software that mapped a hierarchical representation of data to a relational representation of data and synchronized changes made to the data in one representation with the data in the other representation was implemented and executed on computers at Microsoft Corporation in Redmond, WA.


3) Exhibit A is a redacted copy of a document created by one of the named inventors, Daniel Dedu-Constantin, in March of 2000 and last saved in April of 2000 (*see* Exhibit B). Software as described in the sections of Exhibit A entitled “Mapping,” “Updating ROM: sync-ing XOM” and “Updating XML: sync-ing ROM,” – *i.e.*, software that could map relational data (relational object model, ROM) to hierarchical data (XML object model, XOM) and synchronize changes made to the data in either object model with the other object model – was implemented at Microsoft in Redmond, WA at least as early as April 25, 2000.

4) Exhibit B is a screen shot of the properties of the document Implementation Notes for Data Cache.doc showing that Daniel Dedu-Constantin created the document in March of 2000 and it was last saved on April 25, 2000.

5) Exhibit C is a screen shot of the source code checkin history for the file XmlDataDocument. XmlDataDocument is a component that manages the changes made to both XOM and ROM and updates the destination OM (ROM or XOM) according to the changes made to the source OM. The source code checkin log shows that on June 29, 2000, the XmlDataDocument file was checked in to the source code repository after a change was made to the portion of the source code that synchronized changes in ROM with XOM (referred to as Data event changes in the Comment column).

We hereby declare that all statements made in this Declaration are true to the best of our knowledge and belief. We acknowledge that willful false statements are punishable by fine or imprisonment or both (18 U.S.C. §1001) and may jeopardize the validity of the application or any patent issuing thereon.

Respectfully submitted,

  
\_\_\_\_\_  
Daniel Dedu-Constantin

2/26/2007  
Date

\_\_\_\_\_  
Omri Gazitt

\_\_\_\_\_  
Date

\_\_\_\_\_  
Michael J. Pizzo

\_\_\_\_\_  
Date

**PATENT**

**MS146953.01/MSFTP250US**

**CERTIFICATE OF TRANSMISSION**

I hereby certify that this correspondence (along with any paper referred to as being attached or enclosed) is being submitted via the USPTO EFS Filing System; Mail Stop Amendment; Commissioner for Patents; P.O. Box 1450; Alexandria, VA 22313-1450.

February 28, 2007  
Date:

/Casey Martin/  
Casey Martin

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re patent application of:

Applicant(s): Daniel Dedu-Constantin, *et al.*

Examiner: Te Y. Chen

Serial No: 09/894,653

Art Unit: 2161

Filing Date: June 27, 2001

Title: SYSTEM AND METHOD FACILITATING UNIFIED FRAMEWORK FOR  
STRUCTURED/UNSTRUCTURED DATA

**Mail Stop Amendment  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450**

---

**DECLARATION UNDER 37 C.F.R. §1.131**

---

Dear Sir:

The undersigned declare as follows:

- 1) We are the inventors of the above-identified patent application.

2) At least as early as April 11, 2001 software that mapped a hierarchical representation of data to a relational representation of data and synchronized changes made to the data in one representation with the data in the other representation was implemented and executed on computers at Microsoft Corporation in Redmond, WA.

3) Exhibit A is a redacted copy of a document created by one of the named inventors, Daniel Dedu-Constantin, in March of 2000 and last saved in April of 2000 (*see* Exhibit B). Software as described in the sections of Exhibit A entitled "Mapping," "Updating ROM: sync-ing XOM" and "Updating XML: sync-ing ROM," – *i.e.*, software that could map relational data (relational object model, ROM) to hierarchical data (XML object model, XOM) and synchronize changes made to the data in either object model with the other object model – was implemented at Microsoft in Redmond, WA at least as early as April 25, 2000.

4) Exhibit B is a screen shot of the properties of the document Implementation Notes for Data Cache.doc showing that Daniel Dedu-Constantin created the document in March of 2000 and it was last saved on April 25, 2000.

5) Exhibit C is a screen shot of the source code checkin history for the file XmlDataDocument. XmlDataDocument is a component that manages the changes made to both XOM and ROM and updates the destination OM (ROM or XOM) according to the changes made to the source OM. The source code checkin log shows that on June 29, 2000, the XmlDataDocument file was checked in to the source code repository after a change was made to the portion of the source code that synchronized changes in ROM with XOM (referred to as Data event changes in the Comment column).

We hereby declare that all statements made in this Declaration are true to the best of our knowledge and belief. We acknowledge that willful false statements are punishable by fine or imprisonment or both (18 U.S.C. §1001) and may jeopardize the validity of the application or any patent issuing thereon.

Respectfully submitted,

\_\_\_\_\_  
Daniel Dedu-Constantin

\_\_\_\_\_  
Date

  
\_\_\_\_\_  
Omri Gazitt

2/26/07  
\_\_\_\_\_  
Date

\_\_\_\_\_  
Michael J. Pizzo

\_\_\_\_\_  
Date

**PATENT**

**MS146953.01/MSFTP250US**

**CERTIFICATE OF TRANSMISSION**

I hereby certify that this correspondence (along with any paper referred to as being attached or enclosed) is being submitted via the USPTO EFS Filing System; Mail Stop Amendment; Commissioner for Patents; P.O. Box 1450; Alexandria, VA 22313-1450.

February 28, 2007

Date:

/Casey Martin/

Casey Martin

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re patent application of:

Applicant(s): Daniel Dedu-Constantin, *et al.*

Examiner: Te.Y. Chen

Serial No: 09/894,653

Art Unit: 2161

Filing Date: June 27, 2001

Title: SYSTEM AND METHOD FACILITATING UNIFIED FRAMEWORK FOR  
STRUCTURED/UNSTRUCTURED DATA

Mail Stop Amendment  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

---

**DECLARATION UNDER 37 C.F.R. §1.131**

---

Dear Sir:

The undersigned declare as follows:

- 1) We are the inventors of the above-identified patent application.

- 2) At least as early as April 11, 2001 software that mapped a hierarchical representation of data to a relational representation of data and synchronized changes made to the data in one representation with the data in the other representation was implemented and executed on computers at Microsoft Corporation in Redmond, WA.
- 3) Exhibit A is a redacted copy of a document created by one of the named inventors, Daniel Dedu-Constantin, in March of 2000 and last saved in April of 2000 (*see* Exhibit B). Software as described in the sections of Exhibit A entitled "Mapping," "Updating ROM: sync-ing XOM" and "Updating XML: sync-ing ROM," – *i.e.*, software that could map relational data (relational object model, ROM) to hierarchical data (XML object model, XOM) and synchronize changes made to the data in either object model with the other object model – was implemented at Microsoft in Redmond, WA at least as early as April 25, 2000.
- 4) Exhibit B is a screen shot of the properties of the document Implementation Notes for Data Cache.doc showing that Daniel Dedu-Constantin created the document in March of 2000 and it was last saved on April 25, 2000.
- 5) Exhibit C is a screen shot of the source code checkin history for the file XmlDataDocument. XmlDataDocument is a component that manages the changes made to both XOM and ROM and updates the destination OM (ROM or XOM) according to the changes made to the source OM. The source code checkin log shows that on June 29, 2000, the XmlDataDocument file was checked in to the source code repository after a change was made to the portion of the source code that synchronized changes in ROM with XOM (referred to as Data event changes in the Comment column).



We hereby declare that all statements made in this Declaration are true to the best of our knowledge and belief. We acknowledge that willful false statements are punishable by fine or imprisonment or both (18 U.S.C. §1001) and may jeopardize the validity of the application or any patent issuing thereon.

Respectfully submitted,

\_\_\_\_\_  
Daniel Dedu-Constantin

\_\_\_\_\_  
Date

\_\_\_\_\_  
Omri Gazitt

\_\_\_\_\_  
Date

  
\_\_\_\_\_  
Michael J. Pizzo

2/27/2007  
Date

# **Implementation notes for Data Cache**



## Mapping

The following describes how we determine if what is the node category.

- if node is an element and it's identity (name and uri) are the same as the xml identity of one of the tables in the DataSet, then the node is a DataRowNode. DataRowNodes have `_schema == DataTable` that they are in relationship w/. There are special rules for what happens if the node is created outside of the main tree (i.e. disconnected) – see below.
- if the node is under a DataRowNode and the node identity matches one of the xml identities of the columns from the table associated w/ the containing

DataRowNode, then the node is a potential FieldNode. The first [in the tree order (i.e. parent, children, next siblings)] potential FieldNode node is the FieldNode node, and it is the one that gets mapped; it has `_schema == DataColumn`. As a side effect of this, when moving nodes around (or changing their relative position) or adding/removing nodes may re-map FieldNode even if the original FieldNode was not changed. See example below.

- if the node is a text node and it's **direct parent** is a FieldNode and it has no previous sibling or all previous siblings are ValueNodes then the text node is a ValueNode. The value of a field is given by catenation of all the ValueNodes in order they appear (they should be consecutive).

The whole processing happens as if the entity reference nodes have been expanded and have been replaced with their expansion, although in this case the store will most likely have a lot of nodes foliated (since we need to remember that there was an EntityRef there rather than it's expanded context).

Aside from the above mapping, each table has a special column (only one per table) that has TextOnly property true (i.e. `DataColumn.TextOnly == true`). This column holds the value for the catenation of the first consecutive text nodes that appear before any other node types, under the DataRowNode (see `ElementNode.Foliate` function).

Also there each table has a special column which has ghost set to true (`DataColumn.Ghost == true`) this column holds, at the ROM level, the containment relationship present in the XML file (i.e. if we have in XML orders under customer, the order table will have a parent DataRelationship w/ customers and column(s) will be created for this under the cover if needed). This column is used in a relations that have Nested == true (i.e. `DataRelation.Nested == true`); however a nested relation might use non ghosted columns. Ghosted columns never maps to a node, and they are never outputted in the xml: they only serve to find containment operations at the ROM level.



## Updating ROM: sync-ing XOM

### *Updating field values*

When a field value is updated the following happens:

- if the field value is not null
  - if the field value corresponds to a single text node, the value of that text node is changed to match the field value.
  - if the field value corresponds to more than one single text node (consecutive sibling text nodes) then the first one is updated with the new changes, and subsequent ones are removed.
  - if the column type is string and the new value is the empty string, and there is a text node present, then its value is made the empty string. If no text node is present, nothing is done (i.e. no text node with empty value is created).
- if the field value is null then the node corresponding to the field (i.e. the FieldNode corresponding to the column, not the ValueNode) is searched for its

topmost children that are DataRow nodes, or FieldNodes and those children, with their subtrees, are moved as siblings for the field node whose value is being modified. They are inserted as next siblings of the field node, and their relative order is the same as the one in the original tree (the tree order they had initially is preserved, except that now they are siblings). Then the field node is being removed. Effectively, this removes the field node, it's value node(s) and all it's annotations, but preserves any trees that are under it, which have mapped nodes.

### ***DataRow states and updating***

XML always reflect the latest value in the DataRow as it is visible when getting the value using **myDataRow[myColumn]**.

Also XML does not store multiple set of values, as DataRow (new, old and temp), just the ones that are visible from a DataRow when no version is provided (the algo used by DataRow is: use temp values, if no temp values, then use new ones).

### ***Removing a DataRow***

When a DataRow is removed we take the DataRowNode associated w/ the removed DataRow, and insert as next siblings all it's children that are DataRowNodes. The relative order of the new inserted nodes as siblings is preserved. Then the DataRowNode associated w/ the removed DataRow is removed.

This effectively removes the DataRowNode and it's annotations, all it's FieldNodes and their annotations, and all the ValueNodes.

### ***Inserting a DataRow***

**Inserting at the end of the table (appending)**

**Inserting in the middle of the table (between 2 other DataRows)**

**Inserting at the front of the table (as the first DataRow)**

***Creating a new column into a table***

***Removing a column from a table***

***Creating a table***

***Removing a table***



## Updating XML: sync-ing ROM

### *Creating a node*

Creating a node mapped to a DataRow

Creating a node mapped to a potential FieldNode

Creating a node mapped to a potential ValueNode

### *Inserting a node*

Inserting a node mapped to a DataRow

Inserting a node that can be a potential FieldNode

Inserting a node that can be a potential ValueNode

Inserting a node that is a annotation

### *Inserting a tree*

### *Removing a node*

### *Removing a tree*

### *Changing the value of a node*

Text nodes

## DataRow states and row node states

DataRow has 3 sets of values:

- old: those represents values as they are on the back end (i.e. SQL database)
- new: those represents values that the user changed locally.
- temp: those represents values that the user changed locally but not subject to constraints. EndEdit makes those changes permanent locally (i.e. the temp values are saved into new values).

With some exceptions, XOM will be always in sync with the **new** values; because of this we are not interested in temp and old values (except the exceptions).

DataRow can be in one of the following states:

- Detached: just created but not in the rows collection, OR after a committed Delete (hard-delete). DataRow.rowID is -1 in this state. Usually temp value is
- Unchanged: in the rows collection, but not changed from the back end (new == old).
- Modified: in the rows collection, but w/ values changed from the back end (new != old). AKA pending changed

- New: in the rows collection; the row was inserted locally by the user (no corresponding row to the back end): old == -1, new != -1. AKA pending insert
- Deleted: in the rows collection, the row is deleted locally but not on the back end (old != -1, new == -1). AKA pending delete

Once the local changes are saved to the back end (or AcceptChanges is called):

- the pending deleted rows become Detached
- all other rows become Unchanged.

On the XOM side the row node can only be in 2 states:

- in the main (document) tree
- in a fragment (it still belongs to the document).

The following is the mapping between the data row (row) state and the row node state:

- if a row node is in the main tree, then it's associated w/ a data row that is in the unchanged, modified or new state. The values visible from the XOM corresponds to values seen from the ROM (as **new** values, not as **temp** values). unless the row

- if the row nodes in is in fragment: then it is associated either w/ a detached row (both newly created (AKA detached-created) or hard deleted (AKA detached-hardDeleted) ) OR a deleted row (pending).
  - o When it is associated w/ a created row (pending insert, not in the rows collection, AKA detached-created) the values that are visible are the **temp** values NOT the **new** one. This is because a created not-in-collection row has only temp values and there are no new values).

The row node has ElementNode.\_row != null and pointing to the created not-in-collection data row. As a side effect of this we **may** need to sync again the values when the data row is inserted in the collection (if ROM decides to change field values).

- o When it is associated w/ a pending delete, ElementNode.\_row != null and points to the deleted row. Any changes made to the region are not visible in the ROM (the node is pending deleted). If the data row gets rolled back, then the content of the region may change (i.e. new elements/attr/textnodes getting removed/added, text nodes value change, etc). Committing the pending delete row (by AcceptChanges it) will transition this node into detached-hardDeleted state. Moving the row node to the main tree is considered an undo of deletion and reposition of the data row in a new position in the collection (i.e. it will not have the same index).

- When it is associated w/ a hard deleted (detached-hardDeleted), the row node has a null associated data row (i.e. `Element._row == null`). Any changes to XOM are not reflected in any way to ROM. Moving the row node in the main tree will associate it w/ another data row.

Implementation notes for Data Lake doc Properties

General | Custom | Summary

Property	Value
<input type="checkbox"/> Word Count	3098
<input type="checkbox"/> Character Count	17662
<input type="checkbox"/> Line Count	147
<input type="checkbox"/> Paragraph Count	35
<input type="checkbox"/> Scale	No
<input type="checkbox"/> Links Dirty?	0
<input checked="" type="checkbox"/> Comments	

**Origin**

<input checked="" type="checkbox"/> Author	Daniel D.C.
<input type="checkbox"/> Last Saved By	Daniel D.C.
<input checked="" type="checkbox"/> Revision Number	122
<input checked="" type="checkbox"/> Application Name	Microsoft Word 9.0
<input checked="" type="checkbox"/> Company	Microsoft Corp.
<input type="checkbox"/> Date Created	03/30/00 12:54 PM
<input checked="" type="checkbox"/> Date Last Saved	04/25/00 1:09 AM
<input type="checkbox"/> Last Printed	03/31/00 9:49 PM
<input type="checkbox"/> Edit time	01/01/01 6:40 AM

<< Simple

OK Cancel Apply

Rev	Change	Op	Date	Dev	Comment
37	37 Fx:XmlDataDocument.cs	copy	2001/02/01 10:34:09	daniield	Rev Integration Webdata to Main
36	36 Fx:XmlDataDocument.cs	copy	2001/01/26 16:30:16	yanl	webdata -> main integration
35	35 Fx:XmlDataDocument.cs	copy	2001/01/18 01:47:51	ldai	WebData->FXMain integration
34	34 Fx:XmlDataDocument.cs	copy	2001/01/12 19:45:06	nithyas	Reverse Integration
33	33 Fx:XmlDataDocument.cs	copy	2001/01/05 23:40:59	jasonzhu	Reverse integration: webdata -> main
32	32 Fx:XmlDataDocument.cs	copy	2000/12/29 15:05:51	sdub	WebData -> Main integration
31	31 Fx:XmlDataDocument.cs	edit	2000/12/20 00:08:20	jeffdan	Doc comment extraction
30	30 Fx:XmlDataDocument.cs	copy	2000/12/19 12:40:08	neetur	Integration: Webdata -> Main
29	29 Fx:XmlDataDocument.cs	copy	2000/12/07 22:37:57	jruiz	1100 Reverse Integration
28	28 Fx:XmlDataDocument.cs	edit	2000/11/22 00:04:00	chrisan	compatible changes with 9055 compilers...
27	27 Fx:XmlDataDocument.cs	copy	2000/11/21 23:42:00	blained	webdata -> fx live integration
26	26 Fx:XmlDataDocument.cs	copy	2000/11/17 09:25:18	nzung	webdata reverse integration
25	25 Fx:XmlDataDocument.cs	copy	2000/11/13 22:53:41	sreeramn	Reverse-integration from 2216.X integration.
24	24 Fx:XmlDataDocument.cs	copy	2000/11/10 20:00:05	enzol	enzol: Webdata -> Main reverse integration
23	23 Fx:XmlDataDocument.cs	copy	2000/10/25 16:05:33	markash	webdata->main integration
22	22 Fx:XmlDataDocument.cs	copy	2000/10/20 15:26:22	ldai	WebData -> Main Integration
21	21 Fx:XmlDataDocument.cs	copy	2000/10/17 23:58:27	ldai	WebData branch --> Main branch integration
20	20 Fx:XmlDataDocument.cs	copy	2000/10/09 23:23:23	briangru	Reverse-integrate BCL changes from BCLFork to Live FX tree. --
19	19 Fx:XmlDataDocument.cs	copy	2000/10/06 21:37:27	jasonzhu	reverse integration: webdata -> main
18	18 Fx:XmlDataDocument.cs	copy	2000/09/30 13:31:04	neetur	Integration
17	17 Fx:XmlDataDocument.cs	copy	2000/09/26 15:21:29	neetur	integration:
16	16 Fx:XmlDataDocument.cs	copy	2000/09/07 21:31:34	daxh	webdata->main integration
15	15 Fx:XmlDataDocument.cs	copy	2000/08/31 17:51:45	joycec	Webdata->Main integration.
14	14 Fx:XmlDataDocument.cs	copy	2000/08/29 12:57:21	martmaly	Webdata -> Main reverse integration
13	13 Fx:XmlDataDocument.cs	copy	2000/08/26 12:57:30	nzung	webdata->main reverse integrate
12	12 Fx:XmlDataDocument.cs	copy	2000/08/23 00:30:13	nzung	webdata->main integration
11	11 Fx:XmlDataDocument.cs	copy	2000/08/17 20:48:08	daxh	webdata->main integration, includes XSD change
10	10 Fx:XmlDataDocument.cs	copy	2000/08/14 14:10:38	martmaly	Webdata -> Main integration
9	9 Fx:XmlDataDocument.cs	copy	2000/08/07 10:36:59	yanl	reverse integration form webdata
8	8 Fx:XmlDataDocument.cs	copy	2000/08/03 16:47:11	yanl	integration from webdata
7	7 Fx:XmlDataDocument.cs	copy	2000/08/02 16:05:18	yanl	reverse integration from Webdata branch
6	6 Fx:XmlDataDocument.cs	copy	2000/08/01 10:45:30	a-dhyatt	DNA UE Doc Comment Edits
5	5 Fx:XmlDataDocument.cs	copy	2000/07/27 01:23:01	daniield	Webdata checkin (reverse integration from //depot/fx/webdata branch).
4	4 Fx:XmlDataDocument.cs	copy	2000/07/24 23:52:03	daniield	Reverse integrate from WebData to Main branch.
3	3 Fx:XmlDataDocument.cs	copy	2000/07/22 18:09:57	daniield	Reverse integration from WebData branch.
2	2 Fx:XmlDataDocument.cs	copy	2000/07/11 13:14:15	govindad	reverse integrate changes from webdata branch into main
1	1 Fx:XmlDataDocument.cs	branch	2000/06/29 14:02:28	daniield	Reverse integrate changes from webdata branch into main

